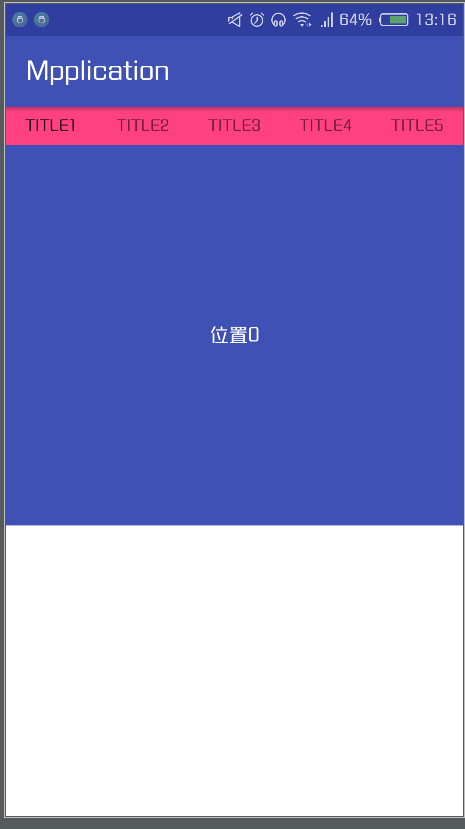
FragmentPagerAdapter

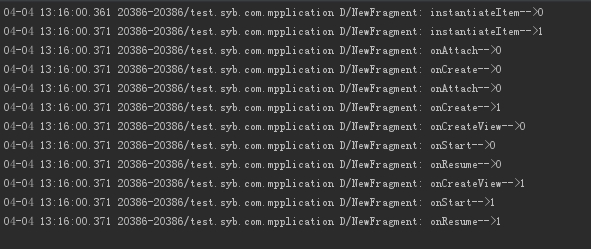
[FragmentPagerAdapter](http://developer.android.com/reference/android/support/v13/app/FragmentPagerAdapter.html) 继承自 PagerAdapter。相比通用的 PagerAdapter，该类更专注于每一页均为 Fragment 的情况。如文档所述，**该类内的每一个生成的 Fragment 都将保存在内存之中**，因此适用于那些相对静态的页，数量也比较少的那种；如果需要处理有很多页，并且数据动态性较大、占用内存较多的情况，应该使用[FragmentStatePagerAdapter](http://developer.android.com/reference/android/support/v4/app/FragmentStatePagerAdapter.html)。FragmentPagerAdapter 重载实现了几个必须的函数，因此来自 PagerAdapter 的函数，我们只需要实现 getCount()，即可。且，由于 FragmentPagerAdapter.instantiateItem() 的实现中，调用了一个新增的虚函数 getItem()，因此，我们还至少需要实现一个 getItem()。因此，总体上来说，相对于继承自 PagerAdapter，更方便一些。

* [**getItem()**](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.1.1_r1/android/support/v4/app/FragmentPagerAdapter.java#FragmentPagerAdapter.getItem%28int%29)
  + 该类中新增的一个虚函数。函数的目的为生成**新的** Fragment 对象。重载该函数时需要注意这一点。在需要时，该函数将被 instantiateItem() 所调用。
  + 如果需要向 Fragment 对象传递相对**静态的数据**时，我们一般通过 Fragment.setArguments() 来进行，这部分代码应当放到 getItem()。它们只会在新生成 Fragment 对象时执行一遍。
  + 如果需要在生成 Fragment 对象后，将**数据集里面一些动态的数据**传递给该 Fragment，那么，这部分代码不适合放到 getItem() 中。因为当数据集发生变化时，往往对应的 Fragment 已经生成，如果传递数据部分代码放到了 getItem() 中，这部分代码将不会被调用。这也是为什么很多人发现调用 PagerAdapter.notifyDataSetChanged() 后，getItem() 没有被调用的一个原因。
* [instantiateItem()](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.1.1_r1/android/support/v4/app/FragmentPagerAdapter.java#FragmentPagerAdapter.instantiateItem%28android.view.ViewGroup%2Cint%29)
  + 函数中[判断一下](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.1.1_r1/android/support/v4/app/FragmentPagerAdapter.java" \l "90" \t "_blank)要生成的 Fragment 是否已经生成过了，如果生成过了，就使用旧的，[旧的将被 Fragment.attach()](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.1.1_r1/android/support/v4/app/FragmentPagerAdapter.java" \l "95" \t "_blank)；如果没有，[就调用 getItem() 生成一个新的](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.1.1_r1/android/support/v4/app/FragmentPagerAdapter.java" \l "97" \t "_blank)，[新的对象将被 FragmentTransation.add()](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.1.1_r1/android/support/v4/app/FragmentPagerAdapter.java#99)。
  + FragmentPagerAdapter 会将所有生成的 Fragment 对象通过 FragmentManager 保存起来备用，以后需要该 Fragment 时，都会从 FragmentManager 读取，**而不会再次调用 getItem() 方法**。
  + 如果需要在生成 Fragment 对象后，将数据集中的一些数据传递给该 Fragment，这部分代码应该放到这个函数的重载里。在我们继承的子类中，重载该函数，并调用 FragmentPagerAdapter.instantiateItem() 取得该函数返回 Fragment 对象，然后，我们该 Fragment 对象中对应的方法，将数据传递过去，然后返回该对象。
  + 否则，如果将这部分传递数据的代码放到 getItem()中，在 PagerAdapter.notifyDataSetChanged() 后，这部分数据设置代码将不会被调用。
* [**destroyItem()**](http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/4.1.1_r1/android/support/v4/app/FragmentPagerAdapter.java#FragmentPagerAdapter.destroyItem%28android.view.ViewGroup%2Cint%2Cjava.lang.Object%29)
  + 该函数被调用后，会对 Fragment 进行 [FragmentTransaction.detach()](http://developer.android.com/reference/android/support/v4/app/FragmentTransaction.html" \l "detach(android.support.v4.app.Fragment)" \t "_blank)。这里不是 remove()，只是 detach()，因此 Fragment 还在 FragmentManager 管理中，Fragment 所占用的资源不会被释放。

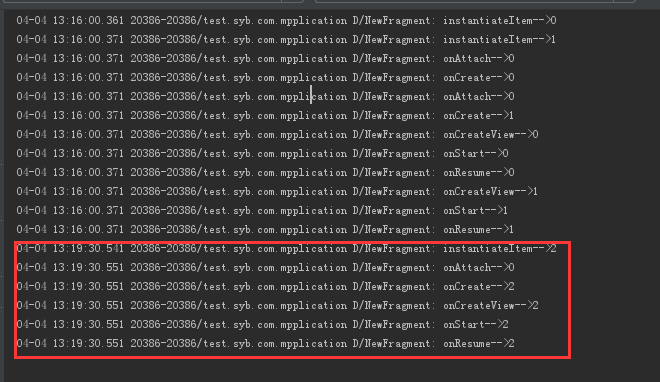
可能都听过ViewPager默认缓存三个子页面；接下来以测试用例来看一下：

、

这是初始运行界面图，通过下面日志可以看到:预加载了与当前页面邻近的页面；



当我滑动到下一个页面即：Title2时：

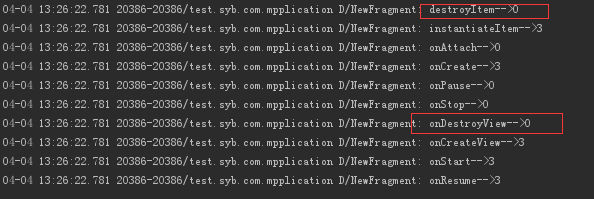


此时可以看到已经预加载到Title3页面。从FragmentPagerAdapter源码可以看出:



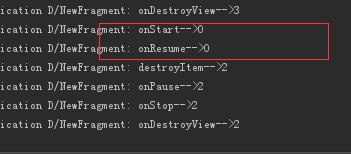
也就是说FragmentPagerAdapter默认是先预加载一页的，比如显示了第1页，就会把第2页也加载了，先调用FragmentPagerAdapter的构造方法 MyPagerAdapter(FragmentManager fm)，再调用instantiateItem(ViewGroup container, int position) 函数中判断一下要生成的 Fragment 是否已经生成过了，如果生成过了，就使用旧的，旧的将被 Fragment.attach()；如果没有，就调用 getItem() 生成一个新的。

当我滑动到Title3时，可以看到日志：



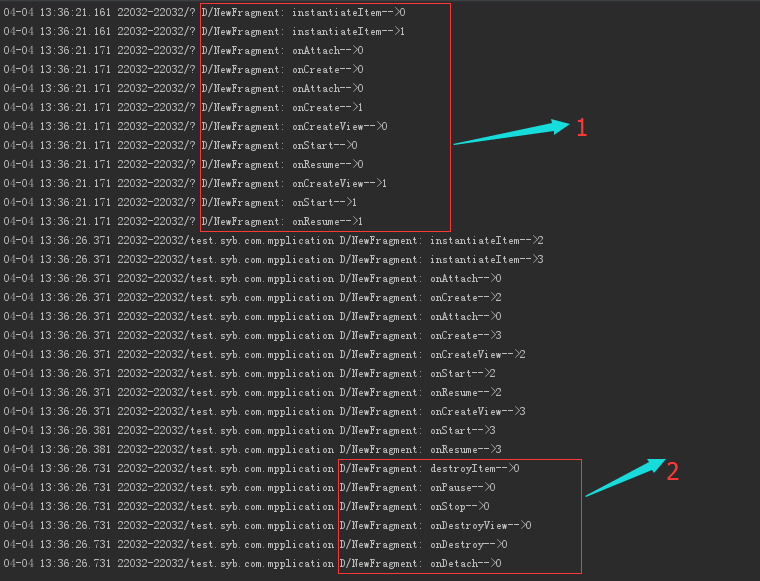
第一个页面只是执行了onDestroyView方法，并没有走onDestroy、onAttach方法；也就是说并没有和当前的activity解绑。

当我这时滑动到第一个页面时，可以看到:

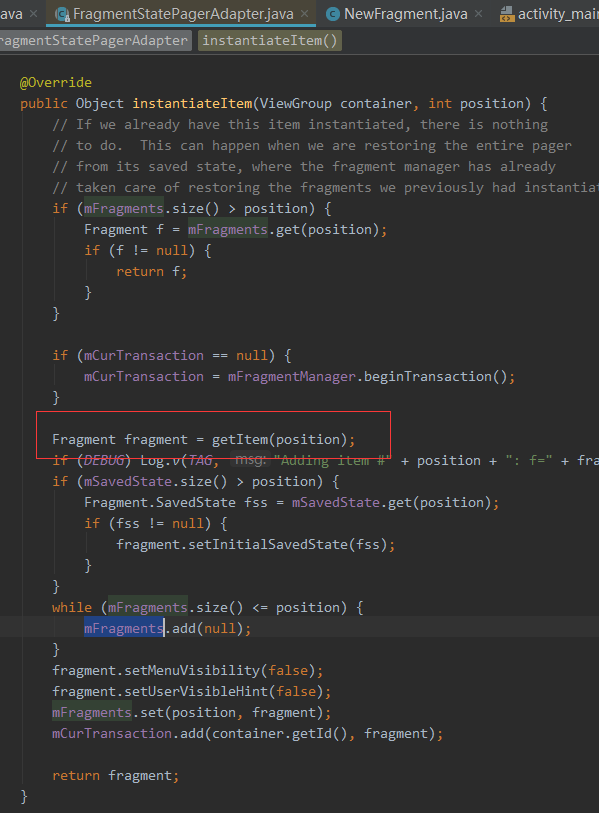


当前的Frament并没有重新走完生命周期方法，只是走了onStart、onResume。

2、这时我再看看使用FragmentStatePagerAdapter重复上面的步骤会出现什么样的结果【我就不一一比较了，就直接贴图说明】：



可以看到【1】是初次初始化时，和FragmentPagerAdapter是一样的，再看[2]我切换到【Title3】时，第一个页面执行了onDestroy、onDetach 方法，与当前activity解绑。



可以看到FragmentStatePagerAdapter源码，并没有像FragmentPagerAdapter中从缓存中取需要展示的Fragment实例。

3 、分析

通过上面的Fragment的生命周期我们可以看到，当ViewPager使用FragmentPagerAdapter时滑动viewpager，Fragment并不会跟acvitity解绑，并且划回该Fragment时，onCreate方法也不会执行。FragmentStatePagerAdapter时，Fragment已经跟activity解绑了，重新划回该Fragment时，Fragment的生命周期会全部走一遍。

因此当我们在Fragment的onCreate方法中执行的网络请求，那么在FragmentPagerAdapter中，该网络请求只会执行一次，而FragmentStatePagerAdapter则会每次都得到执行。